API Write Functionality

The API has been enhanced to allow you to make changes to the **monitoring** and **alerting** states of your monitors.

This is particularly useful if you make use of continuous deployment mechanisms and want to programmatically control your monitors as part of your release cycle.

For those that already make such changes manually via the Portal Settings page, the API will function consistently in that you can change the **monitoring** status of user journeys as well as the **monitoring** and **alerting** status of your page monitors, or steps of user journeys/web service monitors. You can also force the settings for all steps of a user journey/web service with one call.

Please note that as with changing settings in the Portal, if you have any **Scheduled Timings** set up for your monitors, these will continue to operate independently of any changes you make via the API. If you are unsure if you have any set up, please contact Customer Support.

Customers are advised to carefully construct any application or script that is interfacing with the API to ensure their monitors are configured as expected. If a customer's application fails to re-enable monitoring for instance, then you may be running with no monitoring or alerting and not be aware of this.

Each call to make a change will be accounted for as a single request, and is aggregated along with any read requests your company has made each month.

To protect excessive changes in a short period of time to the same monitor, update requests are limited to only allow one change every 5 seconds (per monitor).

Authenticating

In order to be able to make these changes via the API, your company and user account needs to have been granted access as a user of the API service (this is the same as for users needing to read from the API). In addition, your user must be configured as an **Account Administrator** (again, this is consistent with the Portal, as you need this level of access to be able to make changes to monitor settings).

If you wish to make changes via the API, then you need to authenticate slightly differently, to signify that you want to elevate your privileges (you can only do this if you are authorised, as detailed above).

You can either authenticate directly via the URL, or interactively via the API login page (in the same way that you do with the traditional read API). Typically, applications/programs will authenticate via a GET/POST, and whilst testing, users will authenticate via the login page.

The important requirement is that you authenticate with the additional parameter:

GET request
`/Enable/UPDATE`

POST request
`Enable=UPDATE`

For example, authenticating via the URL (GET request) to include write capabilities, you would specify the format:

GET request
/username/*email@address.co.uk***/password/***password***/Enable/UPDATE**

Alternatively, if you are authenticating interactively via the login page, be sure to tick the “Allow updates” box:

Once authenticated, you will be given an API key as normal, however this key will also have permissions to make changes to your monitors via the API.

Note that even if you are an Account Administrator, if you do not authenticate in this way your API key will not be authorised to make changes. If you do this, you must first logout to release your “read only” key, by calling the “Logout” command:

GET request
/*APIkey***/Logout**

Changing Settings

In general, it is only possible to make changes to a single monitor at a time. This will be a change to the user journey, web service, page or step.

The exception is where you can also force a change to all the steps of a user journey or web service monitor.

The important thing to note is that even doing this, you still only specify a single monitor to change (which would be the ‘parent’ user journey or web service monitor). If you try to specify multiple monitors when making an update, you will get an error.

This ensures that when you make a change via the API, you can deal with the response and clearly determine if the change was successfully made or not.

Each of the items documented in the following sections are made by passing additional key/value pairs to the main URL which is made up of the host/domain of the API, your API key and the `/Update` path.

In addition, you must specify the **AccountId** and **Id** of the monitor being changed – these are mandatory.

GET request
/*APIkey*/Update/AccountId/*IdOfAccount*/Id/*IdOfMonitorBeingChanged*/

POST request
URL: **/*APIkey*/Update**
AccountId=*IdOfAccount* **Id=*IdOfMonitorBeingChanged***

Page or Step of a User Journey/Web Service Monitor

It is possible to change both the **monitoring** and **alerting** status for page monitors, as well as steps of a user journey/web service monitor. In fact, when changing the monitoring status for one of these, you must also specify what the alerting status should be also (i.e. you cannot change the monitoring status alone – you must also define if alerting should be enabled or not).

The status is changed by submitting values for the **Monitoring** and **Alerting** keywords, where values can be **TRUE** or **FALSE** (note that **1** and **0** can also be used interchangeably to specify **TRUE** and **FALSE** respectively).

The following lists the different calls that can be made to configure the various states for these types of monitor:

- Enable monitoring and alerting: **/Monitoring/TRUE/Alerting/TRUE**
- Enable monitoring, but disable alerting: **/Monitoring/TRUE/Alerting/FALSE**
- Disable monitoring and alerting: **/Monitoring/FALSE/Alerting/FALSE**

NOTE: If you attempt to just change the monitoring status (for example by only specifying **/Monitoring/TRUE**), you will get an error as you must also define the alerting status. This is only necessary for pages, or steps of a user journey/web service monitor.

As an example, if you have an update API key of 1112223334445556667778889990000AB and want to enable monitoring and disabling alerting for page monitor MN1PG456 in account MN1A123, you would do this as follows:

GET request
/1112223334445556667778889990000AB/Update/AccountId/MN1A123/Id/MN1PG456/Monitoring/TRUE/Alerting/FALSE

POST request
URL: **/1112223334445556667778889990000AB/Update**
AccountId=MN1A123 **Id=MN1PG456** **Monitoring=TRUE** **Alerting=FALSE**

User Journey or Web Service

When configuring a user journey or web service, you define if monitoring should be enabled or not. Note that the alerting status is defined at the step level, so you do not specify an alerting status here. If you want to specifically change the monitoring or alerting status for all steps, then you can force these as detailed in the section below.

The status is changed by submitting values for the **Monitoring** keyword, where values can be **TRUE** or **FALSE**.

The following lists the different calls that can be made to configure the various states for these types of monitor:

- Enable monitoring: `/Monitoring/TRUE`
- Disable monitoring: `/Monitoring/FALSE`

If you also attempt to change the alerting status (for example by only specifying `/Alerting/TRUE`), you will get an error as you cannot change this for user journey/web service monitors. If you want to force the alerting status for all steps, then you can do so as shown in the next section below.

As an example, if you have an update API key of 1112223334445556667778889990000AB and want to enable monitoring for user journey MN1UJ789 in account MN1A123, you would do this as follows:

GET request
/1112223334445556667778889990000AB/Update/AccountId/MN1A123/Id/MN1UJ789/Monitoring/TRUE

POST request
URL: **/1112223334445556667778889990000AB/Update**
AccountId=MN1A123 **Id=MN1UJ789** **Monitoring=TRUE**

Global Change for User Journey or Web Service and all its Steps

It is possible to change a user journey (or web service) monitor as well as all of its steps. This is functionally the same as making such a change in the Portal Settings page (where it will also warn you and force the change to all steps).

As detailed above, the **`/Monitoring/`** key is used to enable/disable the monitoring status of a user journey/web service. However, you can also include the **`/IncludeSteps/`** key to indicate that all steps of the user journey/web service should also be set the same as the monitoring status you are setting.

The important addition is that if you change the monitoring status of a step in this way (globally for the whole user journey/web service), you **must** also define the alerting status for them (as consistent with the approach documented above). This is done by using specifying the **`/StepsAlerting/`** key.

As with all the other parameters documented, the value is set to either **TRUE** or **FALSE**.

The following lists the different calls that can be made to configure the various states for these types of monitor:

- Enable monitoring and alerting for a user journey and all of its steps:
  `/Monitoring/TRUE/IncludeSteps/TRUE/StepsAlerting/TRUE`
- Enable monitoring for a user journey whilst disabling alerting for all of its steps:
  `/Monitoring/TRUE/IncludeSteps/TRUE/StepsAlerting/FALSE`
- Disable monitoring and alerting for a user journey and all of its steps:
  `/Monitoring/FALSE/IncludeSteps/TRUE/StepsAlerting/FALSE`

Note that if you specify not to include steps (**`IncludeSteps/FALSE`**) then the **`StepsAlerting`** value will also be ignored (because you cannot specify just the alerting status for a step – the monitoring status is also needed).

As an example, if you have an update API key of 111222333444555666777888999000AB and want to enable monitoring for user journey MN1UJ789 in account MN1A123, as well as enabling monitoring and disabling alerting for all associated steps, you would do this as follows:

GET request
`/111222333444555666777888999000AB/Update/AccountId/MN1A123/Id/MN1UJ789/Monitoring/TRUE/IncludeSteps/TRUE/StepsAlerting/FALSE`

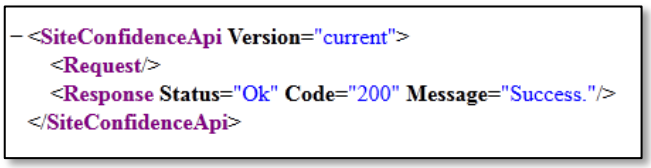
POST request
URL: **`/111222333444555666777888999000AB/Update`**
`AccountId=MN1A123` **`Id=MN1UJ789`** **`Monitoring=TRUE`** **`IncludeSteps=TRUE`** **`StepsAlerting=FALSE`**

Query Responses

Each query made will result in a response that clearly defines if the change was successfully made. Note that changing a value to be the same as the current value is permitted (the query will still result in a successful change).

In addition, you can of course still use the traditional read API to check the value of specific settings if needed. This is particularly useful if you are writing an interface to check the status of the monitor before making a change (e.g. disabling it) so that the original state can be set later on.

The main item to look for in the response is that the `Status` should be `"Ok"` if the change was successfully made:



It is important that any application/script you write to interface with the API checks that updates were successful before proceeding.

Check Status

It is possible to check the **monitoring** and **alerting** status using a standard API query.

For example, to check these for page MN1PG456 in account MN1A123 using an API key of 11122233344455566677788899900AB, you would query as follows:

GET request
`/11122233344455566677788899900AB/Return/[Account[Pages[Page[Monitoring,Alerting]]]]/AccountId/MN1A123/Id/MN1PG456/`

We can see the status for both monitoring and alerting for this page in the response:

In this next example, we can check the **monitoring** status for user journey MN1UJ789 in account MN1A123 using an API key of 1112223334445556667778889990000AB, as follows:

GET request
`/1112223334445556667778889990000AB/Return/[Account[UserJourneys[UserJourney[Monitoring]]]]/AccountId/MN1A5/Id/MN1UJ789/`

We can see the monitoring status for this user journey in the response:

```
- <SiteConfidenceApi Version="current">
    <Request Return="[Account[UserJourneys[UserJourney[Monitoring]]]]" AccountId="MN1A123" Id="MN1UJ789"/>
  - <Response Status="Ok" Code="200" Message="Success.">
    - <Account>
      - <UserJourneys>
          <UserJourney Monitoring="true"/>
        </UserJourneys>
      </Account>
    </Response>
  </SiteConfidenceApi>
```

In this last example, we can query for the **monitoring** status for a user journey, as well as the **monitoring** and **alerting** status of each step, giving us a complete picture of the current settings for this monitor.

For user journey MN1UJ789 which contains the two steps MN1S162 and MN1S163, we can query as follows:

GET request
`/1112223334445556667778889990000AB/Return/[Account[UserJourneys[UserJourney[Monitoring,Steps[Step[Monitoring,Alerting]]]]]]/AccountId/MN1A123/Id/MN1UJ789,MN1S162,MN1S163/`

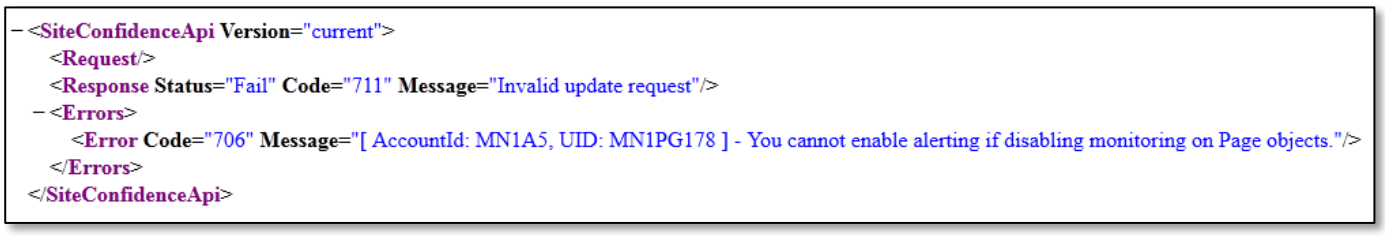
This will give us the following response:

```
- <SiteConfidenceApi Version="current">
    <Request Return="[Account[UserJourneys[UserJourney[Monitoring,Steps[Step[Monitoring,Alerting]]]]]]"
    AccountId="MN1A123" Id="MN1UJ789,MN1S162,MN1S163"/>
  - <Response Status="Ok" Code="200" Message="Success.">
    - <Account>
      - <UserJourneys>
        - <UserJourney Monitoring="true">
          - <Steps>
              <Step Monitoring="true" Alerting="false"/>
              <Step Monitoring="true" Alerting="false"/>
            </Steps>
          </UserJourney>
        </UserJourneys>
      </Account>
    </Response>
  </SiteConfidenceApi>
```

Errors, Warnings and Failures

If an update query fails, you will get an overall `Status` of `"Fail"` along with any further information given as an error code.

For example, if you attempt to update a page monitor by enabling the alerting status whilst disabling the monitoring status (which is an invalid configuration), you would get the following response:



We can see that there was a failure because the update included an invalid update request. In addition, there is a nested `Error Code` element in `Errors` that provides more information as to why the request was invalid.

It is important that any application/script you write to interface with the API checks that updates were successful before proceeding.

The following table lists the details of the various **error** conditions:

Error Code	Description
700	You tried to update multiple monitors in one query (this is not permitted)
701	You specified more than one account ID whilst updating (this is not permitted)
702	You attempted to update a monitor type that is not supported
703	You attempted to update a parameter that is not supported for the type of monitor chosen
704	No value was given against the parameter (set to **TRUE** or **FALSE**)
705	The given value is not supported (set to **TRUE** or **FALSE**)
706	You must specify the **Monitoring** and **Alerting** status for this kind of monitor
707	The monitor you are trying to change is invalid
708	You are attempting to update a monitor with a read only API key
709	No account ID has been specified
710	No monitor ID has been specified
711	The update request is invalid
712	You have attempted to update a monitor setting too soon after you last changed it – wait 5 seconds and try again
713	An internal error has occurred
714	You have specified an invalid parameter
715	The monitor you are trying to change does not exist
716	The account you are trying to change does not exist

In addition, **warning** conditions can occur, which are detailed below:

Warning Code	Description
1100	You have attempted to authenticate with the API and requested to enable updates, however you are not an Account Administrator. You have been granted an API key, but this will be read-only.